PROMINEO TECH

Intro to Java Week 6 Coding Assignment

URL to GitHub Repository: https://github.com/Jeffrweinstein/Week6War

URL to Public Link of your Video: https://www.youtube.com/watch?v=j2Ldnir5b2Q

Instructions:

1. Follow the **Coding Steps** below to complete this assignment.
   - In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed.
   - Create a new repository on GitHub for this week's assignment and push your completed code to this dedicated repo.
   - Create a video showcasing your work:
     - In this video: record and present your project verbally while showing the results of the working project.
     - Easy way to Create a video: Start a meeting in Zoom, share your screen, open Eclipse with the code and your Console window, start recording & record yourself describing and running the program showing the results.
     - Your video should be a maximum of 5 minutes.
     - Upload your video with a public link.
     - Easy way to Create a Public Video Link: Upload your video recording to YouTube with a public link.
2. In addition, please include the following in your Coding Assignment Document:
   - The URL for this week's GitHub repository.
   - The URL of the public link of your video.
3. Save the Coding Assignment Document as a .pdf and do the following:
   - Push the .pdf to the GitHub repo for this week.
   - Upload the .pdf to the LMS in your Coding Assignment Submission.

Intro to Java Week 6 Coding Assignment

Coding Steps — Java Final Project:

For the final project you will be creating an automated version of the classic card game *WAR*.

1. Create the following classes:
   a. Card
      i. Fields
         1. **value** (contains a value from 2-14 representing cards 2-Ace)
         2. **name** (e.g. Ace of Diamonds, or Two of Hearts)
      ii. Methods
         1. Getters and Setters
         2. **describe** (prints out information about a card)
   b. Deck
      i. Fields
         1. **cards** (List of Card)
      ii. Methods
         1. **shuffle** (randomizes the order of the cards)
         2. **draw** (removes and returns the top card of the Cards field)
         3. In the constructor, when a new Deck is instantiated, the Cards field should be populated with the standard 52 cards.
   c. Player
      i. Fields
         1. **hand** (List of Card)
         2. **score** (set to 0 in the constructor)
         3. **name**
      ii. Methods
         1. **describe** (prints out information about the player and calls the describe method for each card in the Hand List)
         2. **flip** (removes and returns the top card of the Hand)
         3. **draw** (takes a Deck as an argument and calls the draw method on the deck, adding the returned Card to the hand field)
         4. **incrementScore** (adds 1 to the Player's score field)

2. Create a class called App with a main method.

   a) Instantiate a Deck and two Players, call the shuffle method on the deck.

   b) Using a traditional for loop, iterate 52 times calling the Draw method on the other player each iteration using the Deck you instantiated.

   c) Using a traditional for loop, iterate 26 times and call the flip method for each player.

   d) Compare the value of each card returned by the two player's flip methods. Call the incrementScore method on the player whose card has the higher value.

   e) After the loop, compare the final score from each player.

   f) Print the final score of each player and either "Player 1", "Player 2", or "Draw" depending on which score is higher or if they are both the same.

3. Tips: Printing out information throughout the game adds value including easier debugging as you progress and a better user experience.

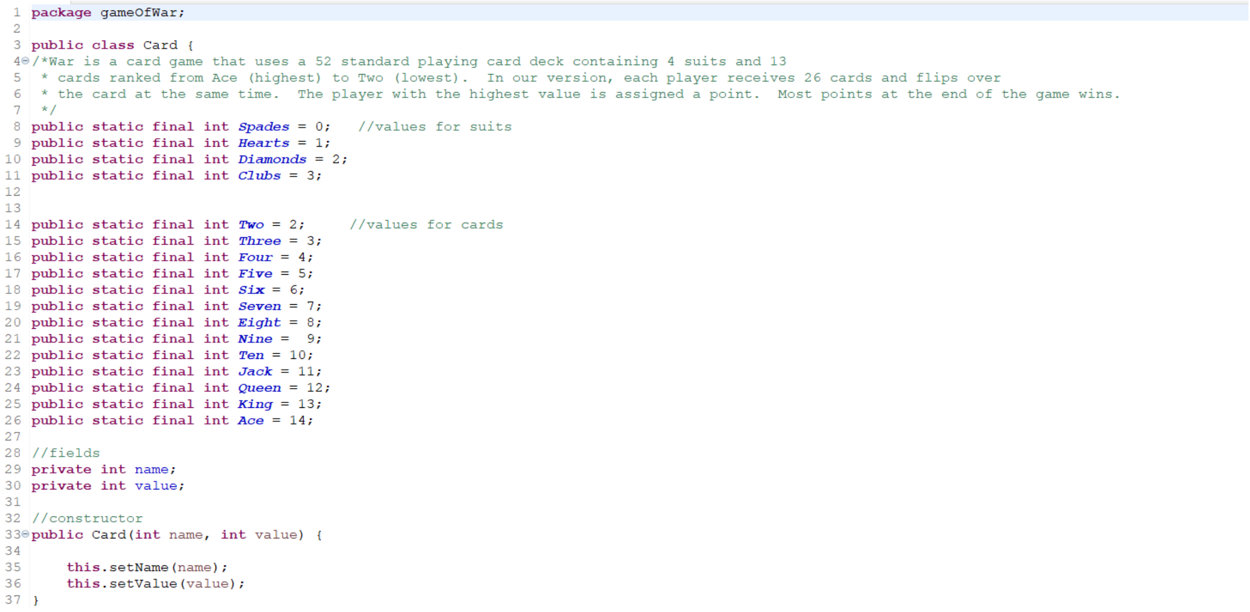
   a) Using the Card describe() method when each card is flipped illustrates the game play.

   b) Printing the winner of each turn adds interest.

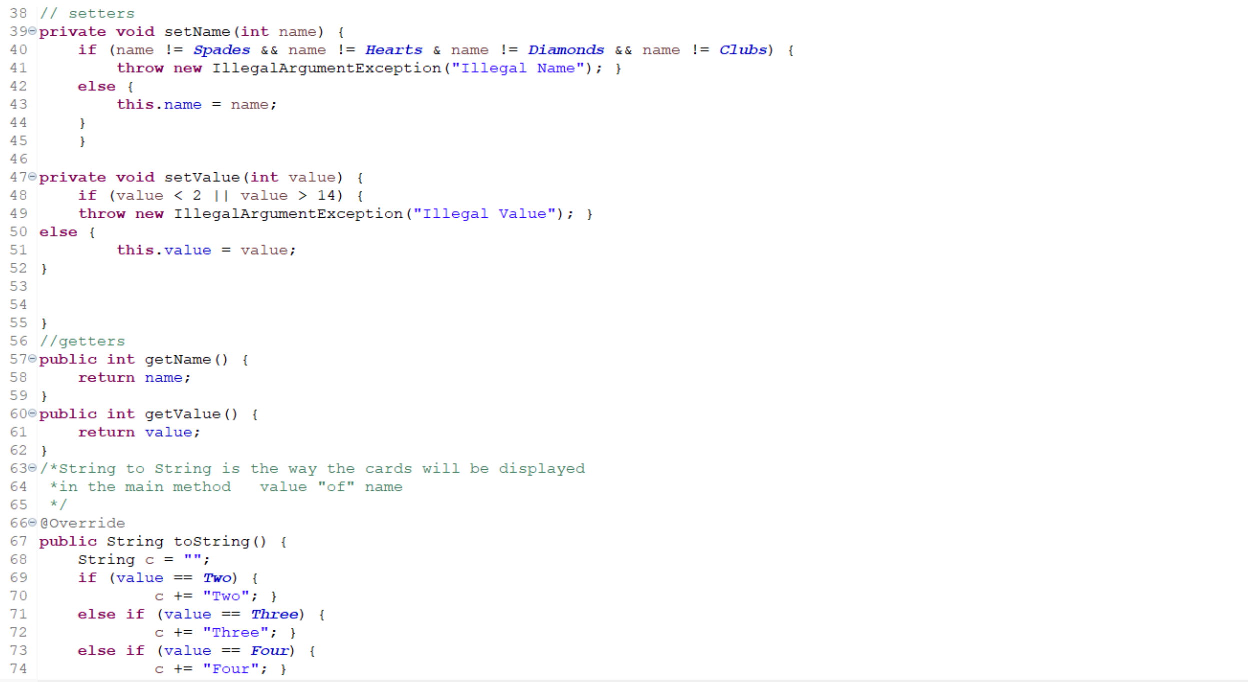
   c) Printing the updated score after each turn shows game progression.

   d) At the end of the game: print the final score of each player and the winner's name or "Draw" if the result is a tie.

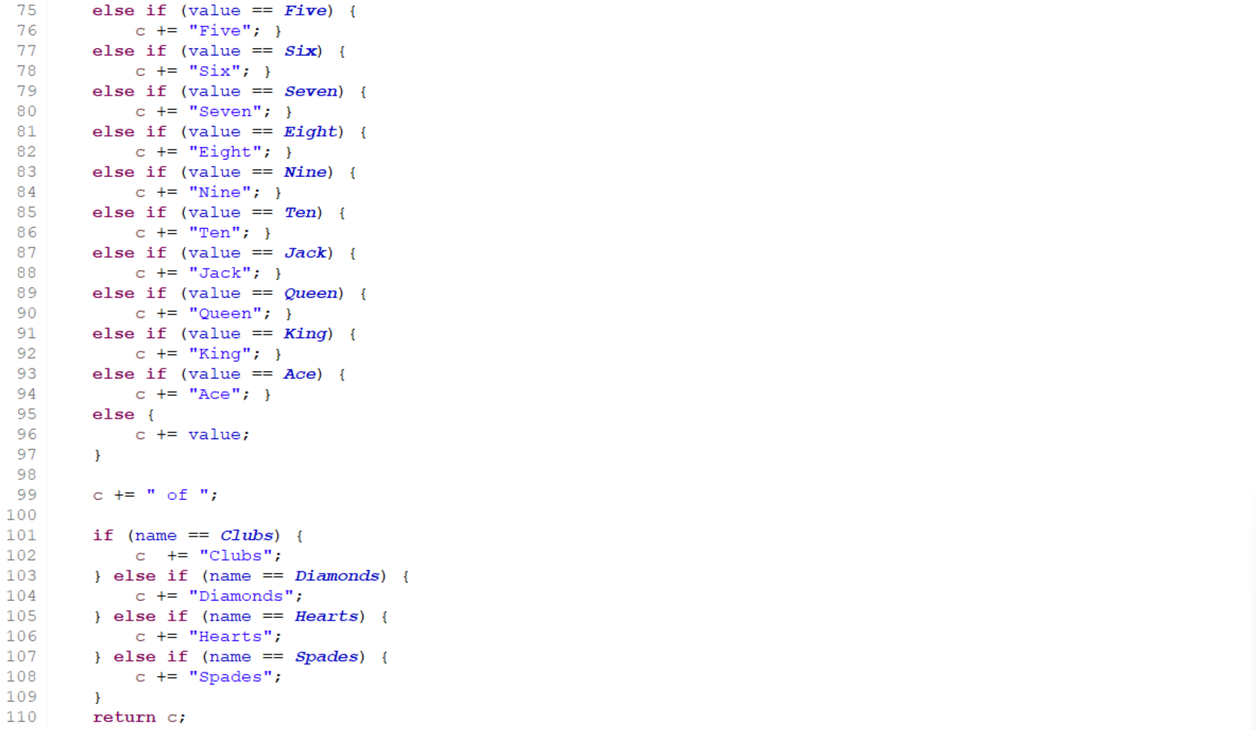
Intro to Java Week 6 Coding Assignment



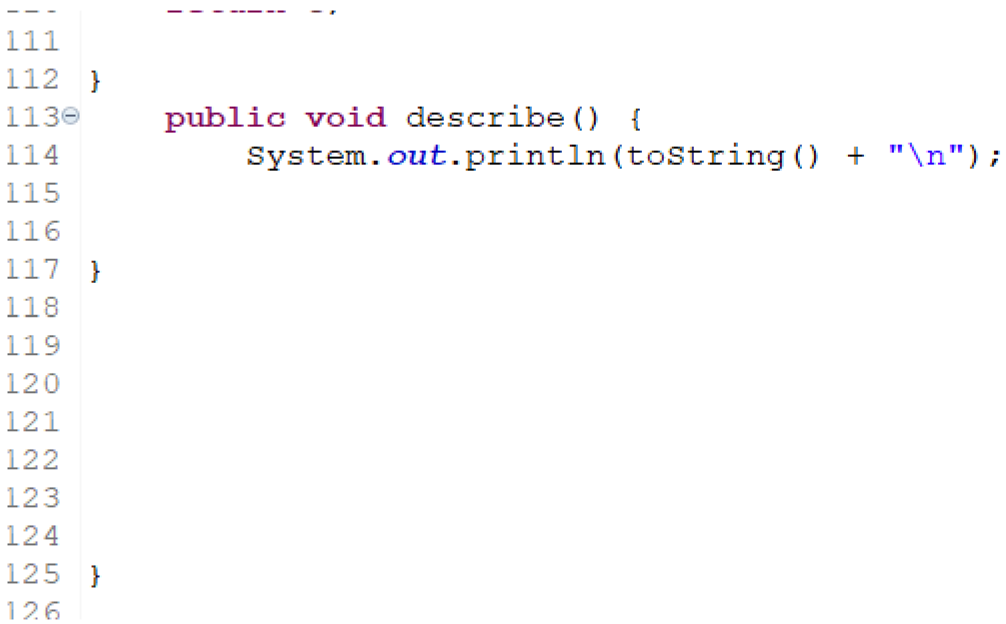
```java
package gameOfWar;

public class Card {
/*War is a card game that uses a 52 standard playing card deck containing 4 suits and 13
 * cards ranked from Ace (highest) to Two (lowest).  In our version, each player receives 26 cards and flips over
 * the card at the same time.  The player with the highest value is assigned a point.  Most points at the end of the game wins.
 */
public static final int Spades = 0;   //values for suits
public static final int Hearts = 1;
public static final int Diamonds = 2;
public static final int Clubs = 3;


public static final int Two = 2;     //values for cards
public static final int Three = 3;
public static final int Four = 4;
public static final int Five = 5;
public static final int Six = 6;
public static final int Seven = 7;
public static final int Eight = 8;
public static final int Nine =  9;
public static final int Ten = 10;
public static final int Jack = 11;
public static final int Queen = 12;
public static final int King = 13;
public static final int Ace = 14;

//fields
private int name;
private int value;

//constructor
public Card(int name, int value) {

    this.setName(name);
    this.setValue(value);
}
```



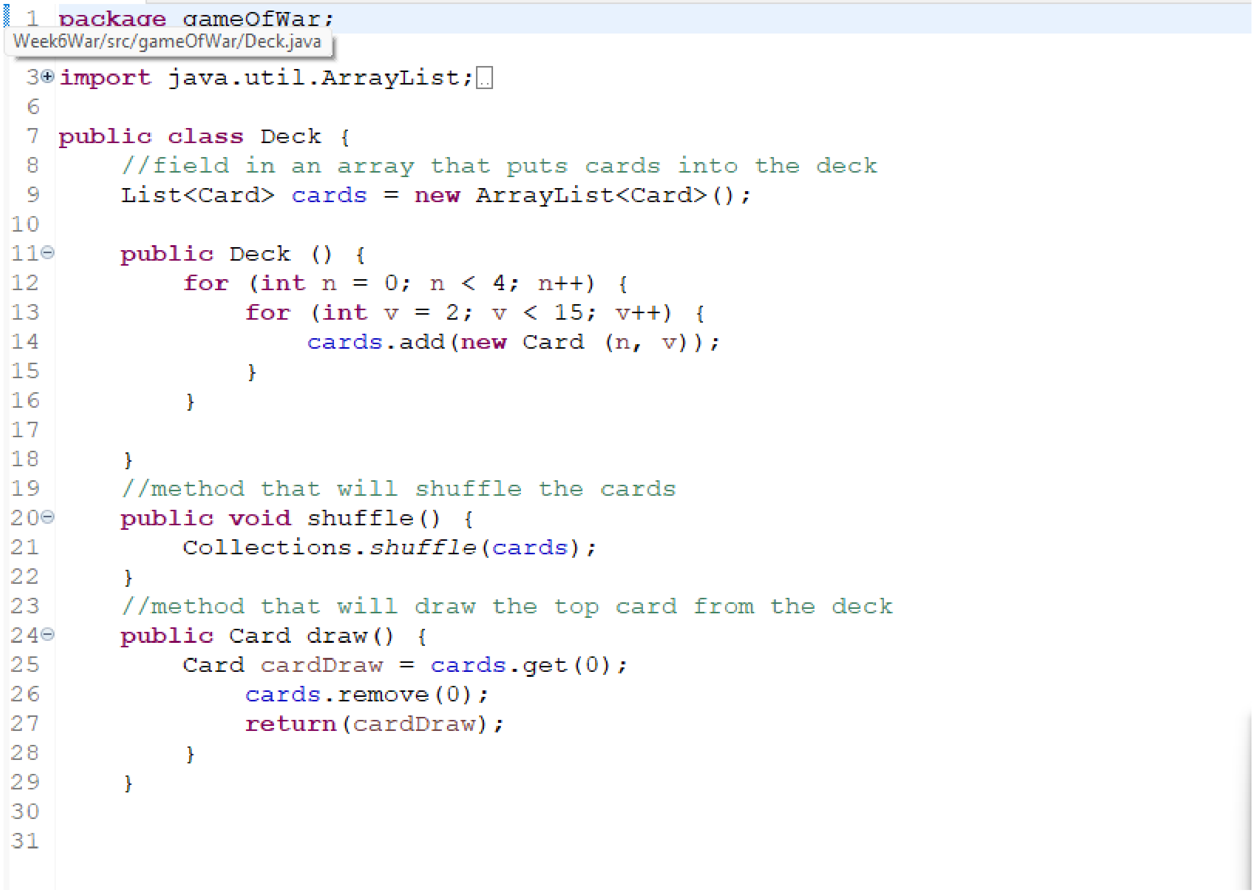
```java
// setters
private void setName(int name) {
    if (name != Spades && name != Hearts & name != Diamonds && name != Clubs) {
        throw new IllegalArgumentException("Illegal Name"); }
    else {
        this.name = name;
    }
    }

private void setValue(int value) {
    if (value < 2 || value > 14) {
    throw new IllegalArgumentException("Illegal Value"); }
else {
        this.value = value;
}



}
//getters
public int getName() {
    return name;
}
public int getValue() {
    return value;
}
/*String to String is the way the cards will be displayed
 *in the main method   value "of" name
 */
@Override
public String toString() {
    String c = "";
    if (value == Two) {
            c += "Two"; }
    else if (value == Three) {
            c += "Three"; }
    else if (value == Four) {
            c += "Four"; }
```



```
    else if (value == Five) {
        c += "Five"; }
    else if (value == Six) {
        c += "Six"; }
    else if (value == Seven) {
        c += "Seven"; }
    else if (value == Eight) {
        c += "Eight"; }
    else if (value == Nine) {
        c += "Nine"; }
    else if (value == Ten) {
        c += "Ten"; }
    else if (value == Jack) {
        c += "Jack"; }
    else if (value == Queen) {
        c += "Queen"; }
    else if (value == King) {
        c += "King"; }
    else if (value == Ace) {
        c += "Ace"; }
    else {
        c += value;
    }

    c += " of ";

    if (name == Clubs) {
        c  += "Clubs";
    } else if (name == Diamonds) {
        c += "Diamonds";
    } else if (name == Hearts) {
        c += "Hearts";
    } else if (name == Spades) {
        c += "Spades";
    }
    return c;
```

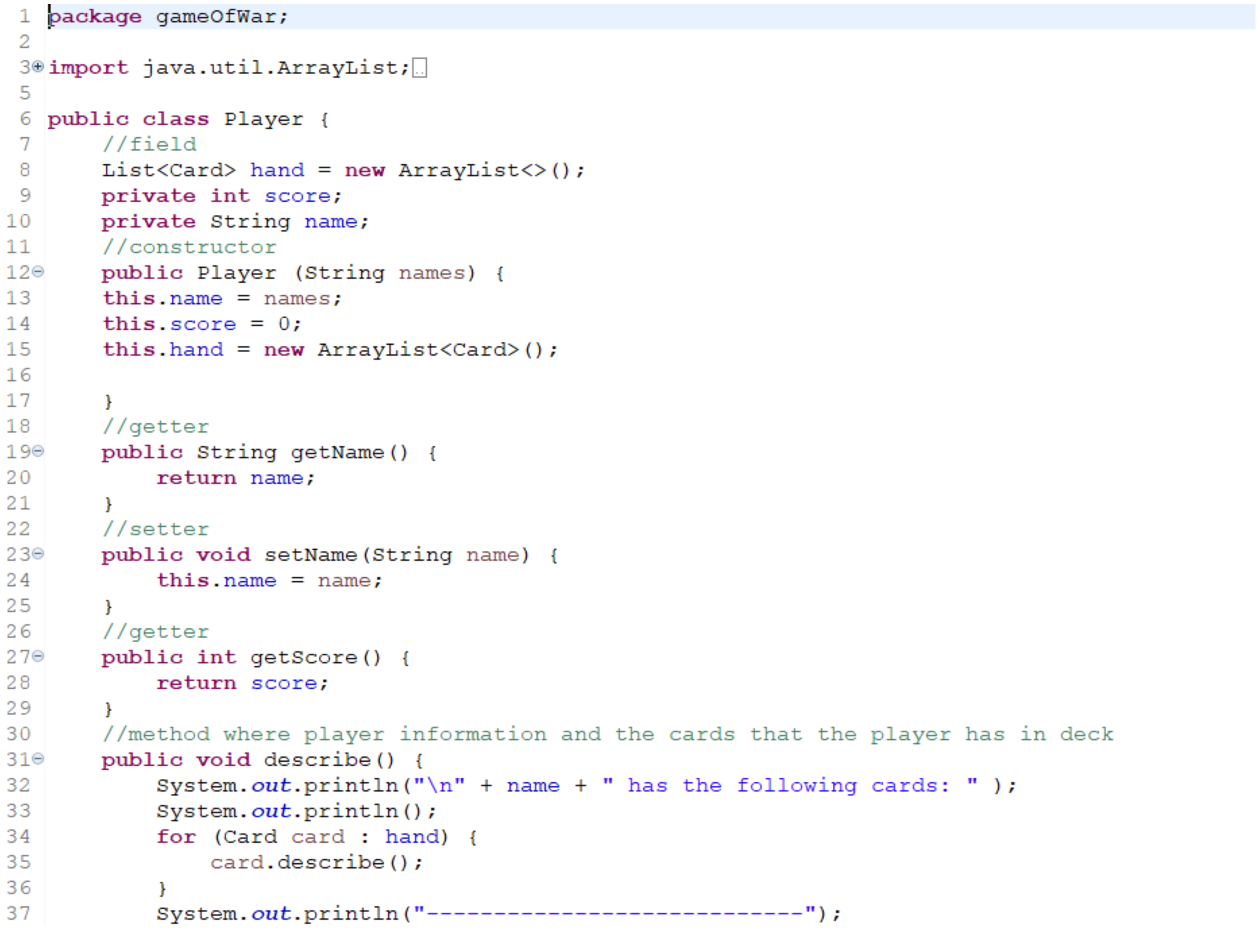


```
}
    public void describe() {
        System.out.println(toString() + "\n");



}







}
```

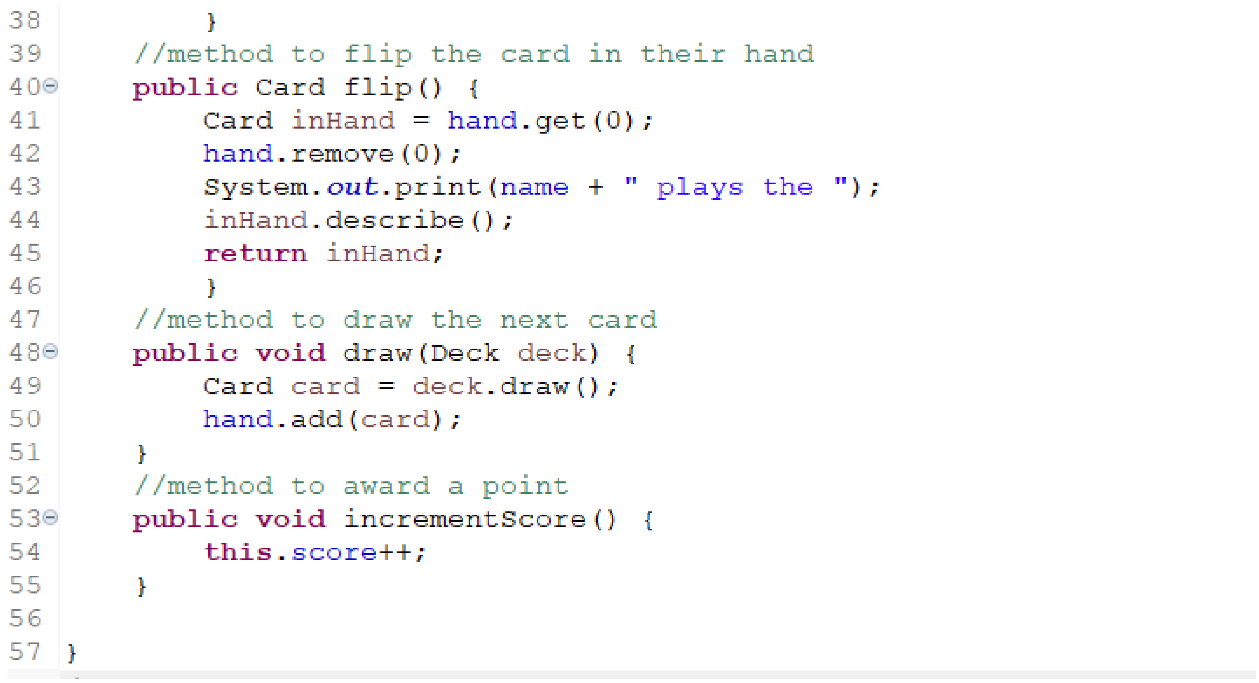


```java
package gameOfWar;

import java.util.ArrayList;

public class Deck {
    //field in an array that puts cards into the deck
    List<Card> cards = new ArrayList<Card>();

    public Deck () {
        for (int n = 0; n < 4; n++) {
            for (int v = 2; v < 15; v++) {
                cards.add(new Card (n, v));
            }
        }


    }
    //method that will shuffle the cards
    public void shuffle() {
        Collections.shuffle(cards);
    }
    //method that will draw the top card from the deck
    public Card draw() {
        Card cardDraw = cards.get(0);
            cards.remove(0);
            return(cardDraw);
        }
    }


```

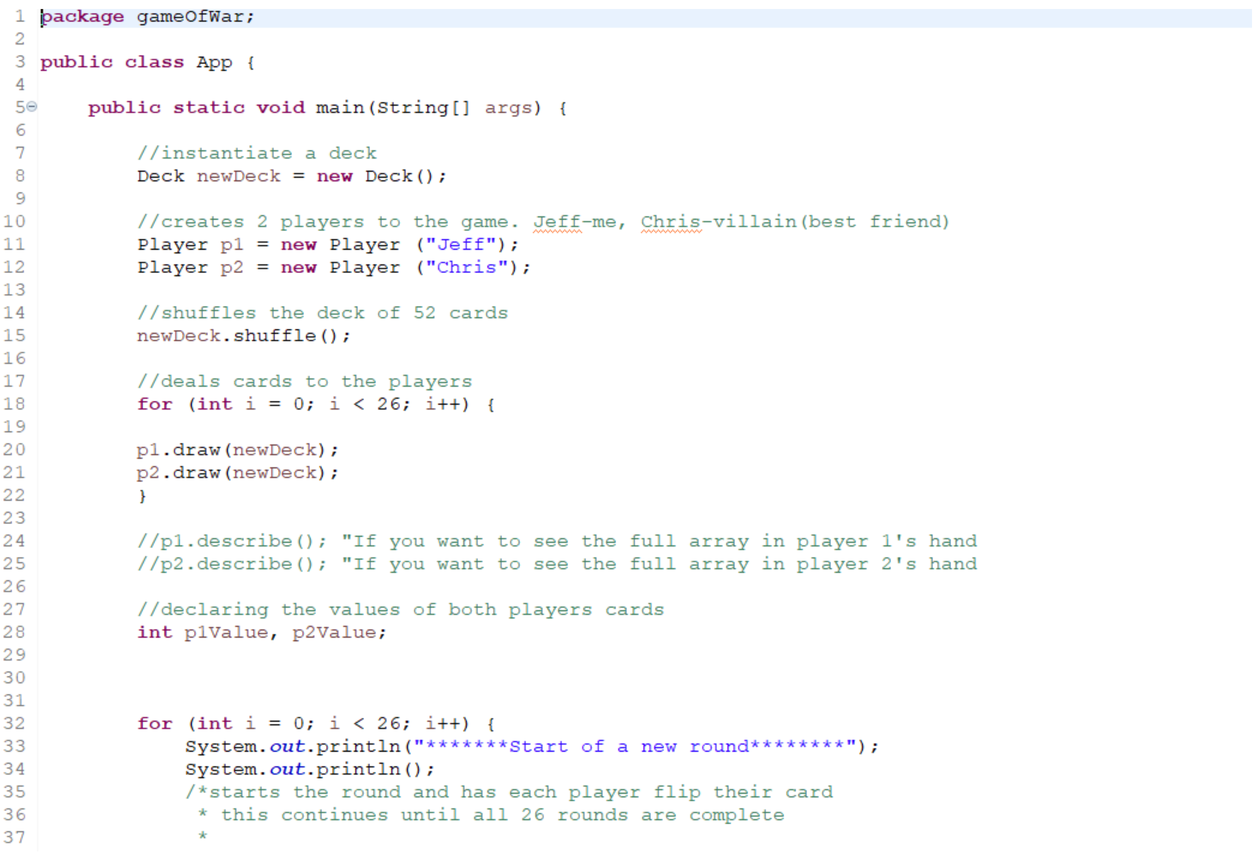
Intro to Java Week 6 Coding Assignment



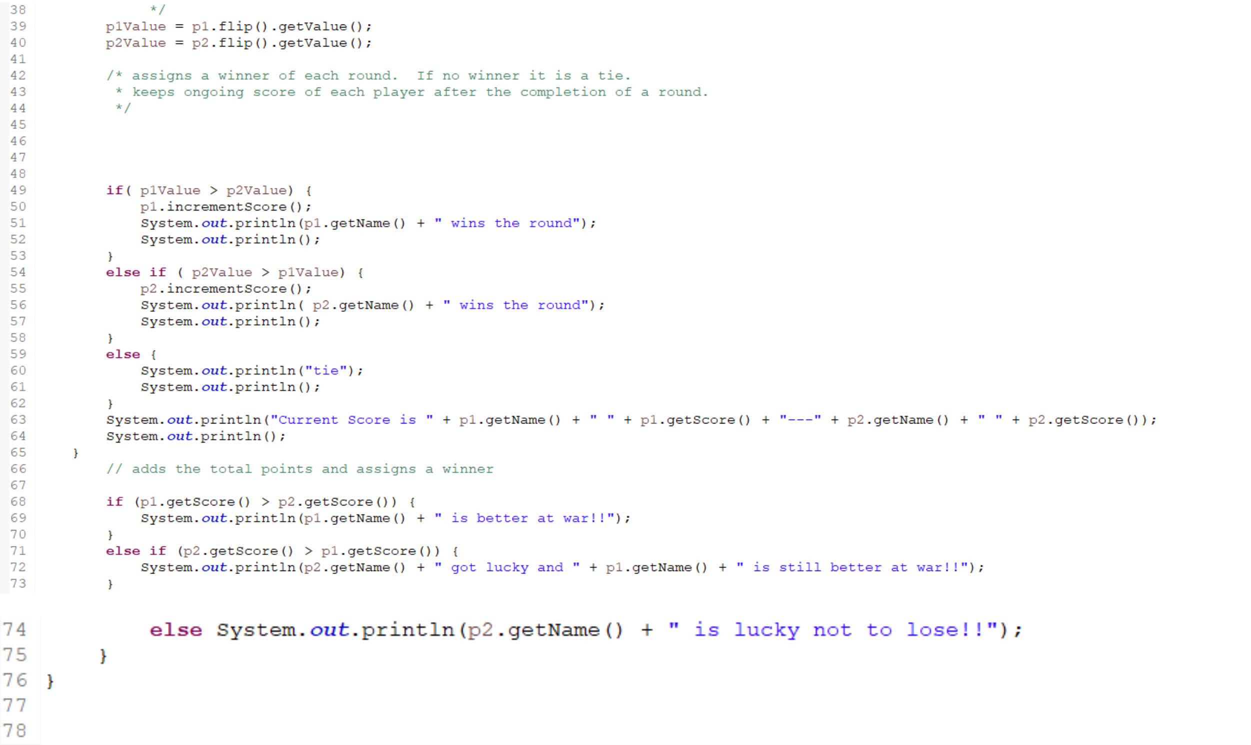
```
package gameOfWar;

import java.util.ArrayList;

public class Player {
    //field
    List<Card> hand = new ArrayList<>();
    private int score;
    private String name;
    //constructor
    public Player (String names) {
    this.name = names;
    this.score = 0;
    this.hand = new ArrayList<Card>();

    }
    //getter
    public String getName() {
        return name;
    }
    //setter
    public void setName(String name) {
        this.name = name;
    }
    //getter
    public int getScore() {
        return score;
    }
    //method where player information and the cards that the player has in deck
    public void describe() {
        System.out.println("\n" + name + " has the following cards: " );
        System.out.println();
        for (Card card : hand) {
            card.describe();
        }
        System.out.println("----------------------------");
```



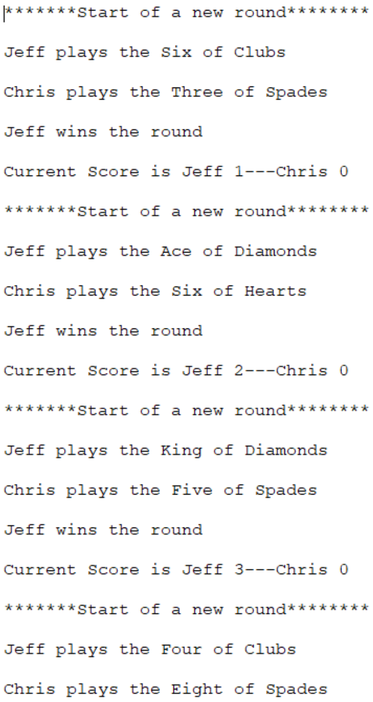
```java
        }
    //method to flip the card in their hand
    public Card flip() {
        Card inHand = hand.get(0);
        hand.remove(0);
        System.out.print(name + " plays the ");
        inHand.describe();
        return inHand;
        }
    //method to draw the next card
    public void draw(Deck deck) {
        Card card = deck.draw();
        hand.add(card);
    }
    //method to award a point
    public void incrementScore() {
        this.score++;
    }

}
```



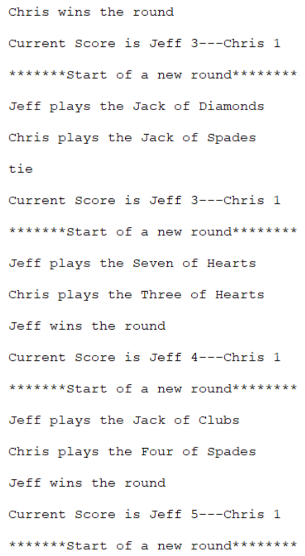
```java
package gameOfWar;

public class App {

    public static void main(String[] args) {

        //instantiate a deck
        Deck newDeck = new Deck();

        //creates 2 players to the game. Jeff-me, Chris-villain(best friend)
        Player p1 = new Player ("Jeff");
        Player p2 = new Player ("Chris");

        //shuffles the deck of 52 cards
        newDeck.shuffle();

        //deals cards to the players
        for (int i = 0; i < 26; i++) {

        p1.draw(newDeck);
        p2.draw(newDeck);
        }

        //p1.describe(); "If you want to see the full array in player 1's hand
        //p2.describe(); "If you want to see the full array in player 2's hand

        //declaring the values of both players cards
        int p1Value, p2Value;



        for (int i = 0; i < 26; i++) {
            System.out.println("*******Start of a new round********");
            System.out.println();
            /*starts the round and has each player flip their card
             * this continues until all 26 rounds are complete
             *
```



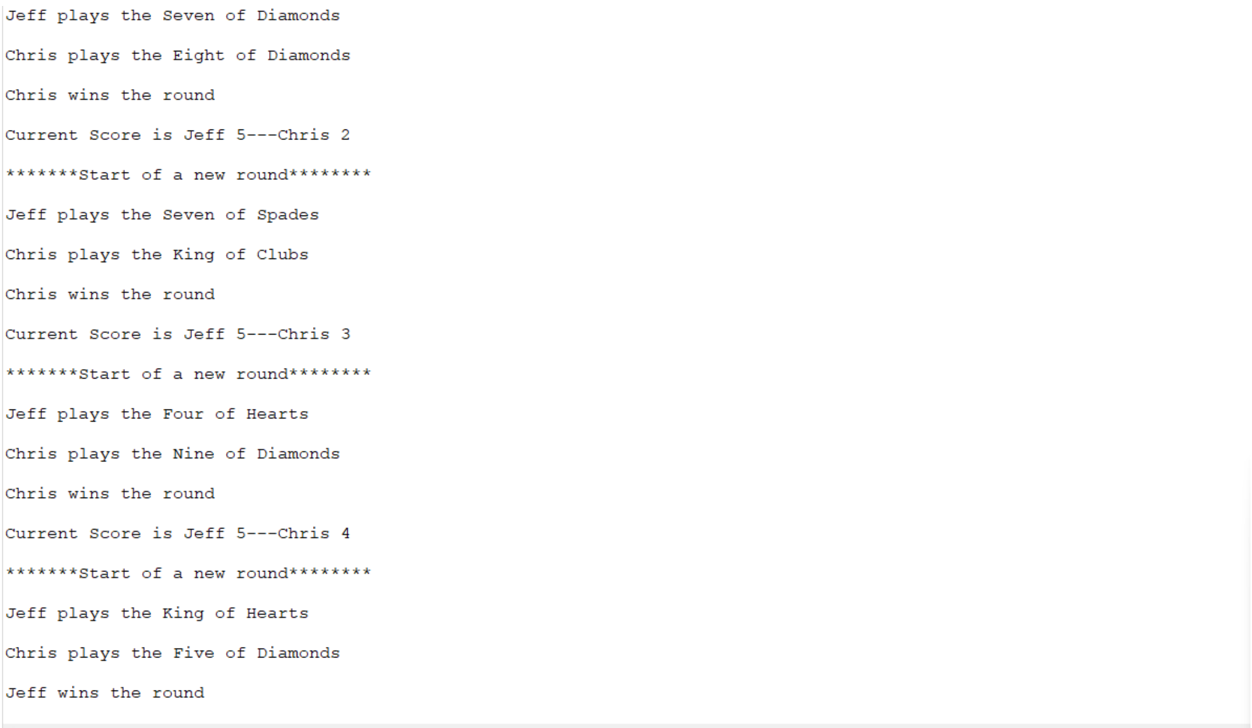
```java
             */
        p1Value = p1.flip().getValue();
        p2Value = p2.flip().getValue();

        /* assigns a winner of each round.  If no winner it is a tie.
         * keeps ongoing score of each player after the completion of a round.
         */




        if( p1Value > p2Value) {
            p1.incrementScore();
            System.out.println(p1.getName() + " wins the round");
            System.out.println();
        }
        else if ( p2Value > p1Value) {
            p2.incrementScore();
            System.out.println( p2.getName() + " wins the round");
            System.out.println();
        }
        else {
            System.out.println("tie");
            System.out.println();
        }
        System.out.println("Current Score is " + p1.getName() + " " + p1.getScore() + "---" + p2.getName() + " " + p2.getScore());
        System.out.println();
    }
        // adds the total points and assigns a winner

        if (p1.getScore() > p2.getScore()) {
            System.out.println(p1.getName() + " is better at war!!");
        }
        else if (p2.getScore() > p1.getScore()) {
            System.out.println(p2.getName() + " got lucky and " + p1.getName() + " is still better at war!!");
        }
        else System.out.println(p2.getName() + " is lucky not to lose!!");
    }
}
```



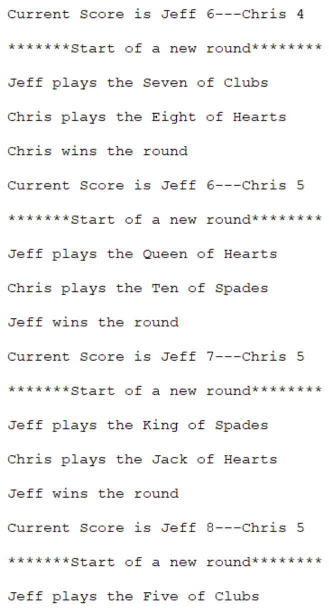
```
*******Start of a new round********

Jeff plays the Six of Clubs

Chris plays the Three of Spades

Jeff wins the round

Current Score is Jeff 1---Chris 0

*******Start of a new round********

Jeff plays the Ace of Diamonds

Chris plays the Six of Hearts

Jeff wins the round

Current Score is Jeff 2---Chris 0

*******Start of a new round********

Jeff plays the King of Diamonds

Chris plays the Five of Spades

Jeff wins the round

Current Score is Jeff 3---Chris 0

*******Start of a new round********

Jeff plays the Four of Clubs

Chris plays the Eight of Spades
```



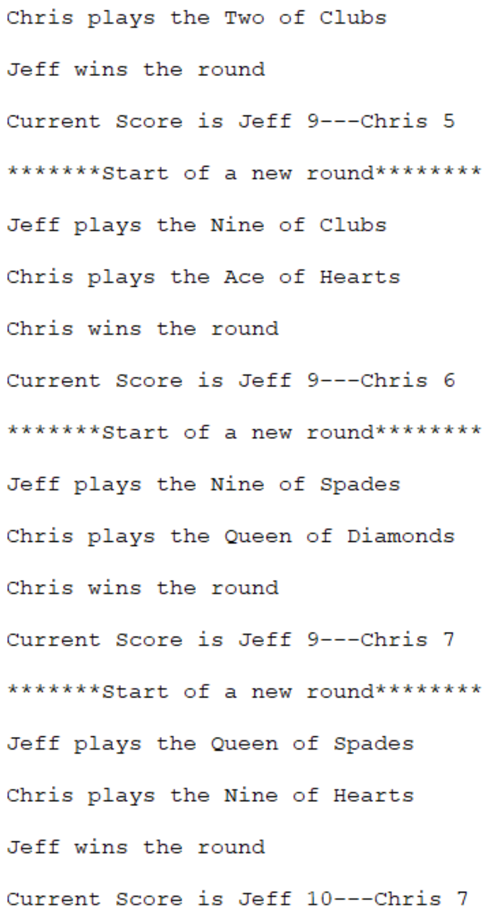
```
Chris wins the round

Current Score is Jeff 3---Chris 1

*******Start of a new round********

Jeff plays the Jack of Diamonds

Chris plays the Jack of Spades

tie

Current Score is Jeff 3---Chris 1

*******Start of a new round********

Jeff plays the Seven of Hearts

Chris plays the Three of Hearts

Jeff wins the round

Current Score is Jeff 4---Chris 1

*******Start of a new round********

Jeff plays the Jack of Clubs

Chris plays the Four of Spades

Jeff wins the round

Current Score is Jeff 5---Chris 1

*******Start of a new round********
```



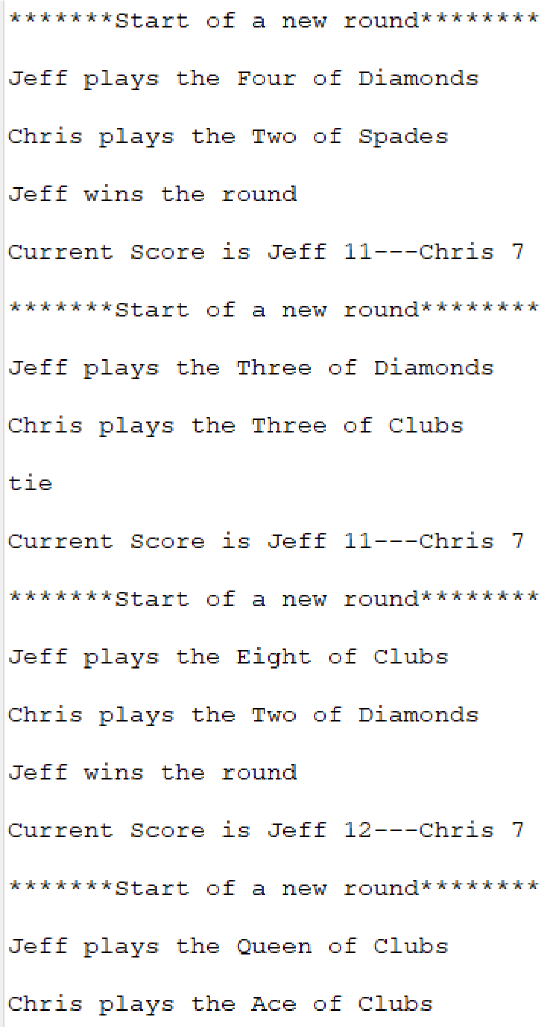
```
Jeff plays the Seven of Diamonds

Chris plays the Eight of Diamonds

Chris wins the round

Current Score is Jeff 5---Chris 2

*******Start of a new round********

Jeff plays the Seven of Spades

Chris plays the King of Clubs

Chris wins the round

Current Score is Jeff 5---Chris 3

*******Start of a new round********

Jeff plays the Four of Hearts

Chris plays the Nine of Diamonds

Chris wins the round

Current Score is Jeff 5---Chris 4

*******Start of a new round********

Jeff plays the King of Hearts

Chris plays the Five of Diamonds

Jeff wins the round
```



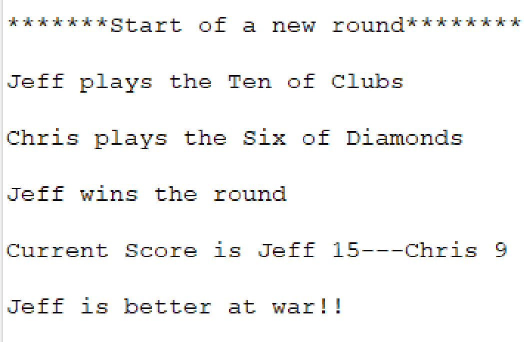
```
Current Score is Jeff 6---Chris 4

*******Start of a new round********

Jeff plays the Seven of Clubs

Chris plays the Eight of Hearts

Chris wins the round

Current Score is Jeff 6---Chris 5

*******Start of a new round********

Jeff plays the Queen of Hearts

Chris plays the Ten of Spades

Jeff wins the round

Current Score is Jeff 7---Chris 5

*******Start of a new round********

Jeff plays the King of Spades

Chris plays the Jack of Hearts

Jeff wins the round

Current Score is Jeff 8---Chris 5

*******Start of a new round********

Jeff plays the Five of Clubs
```



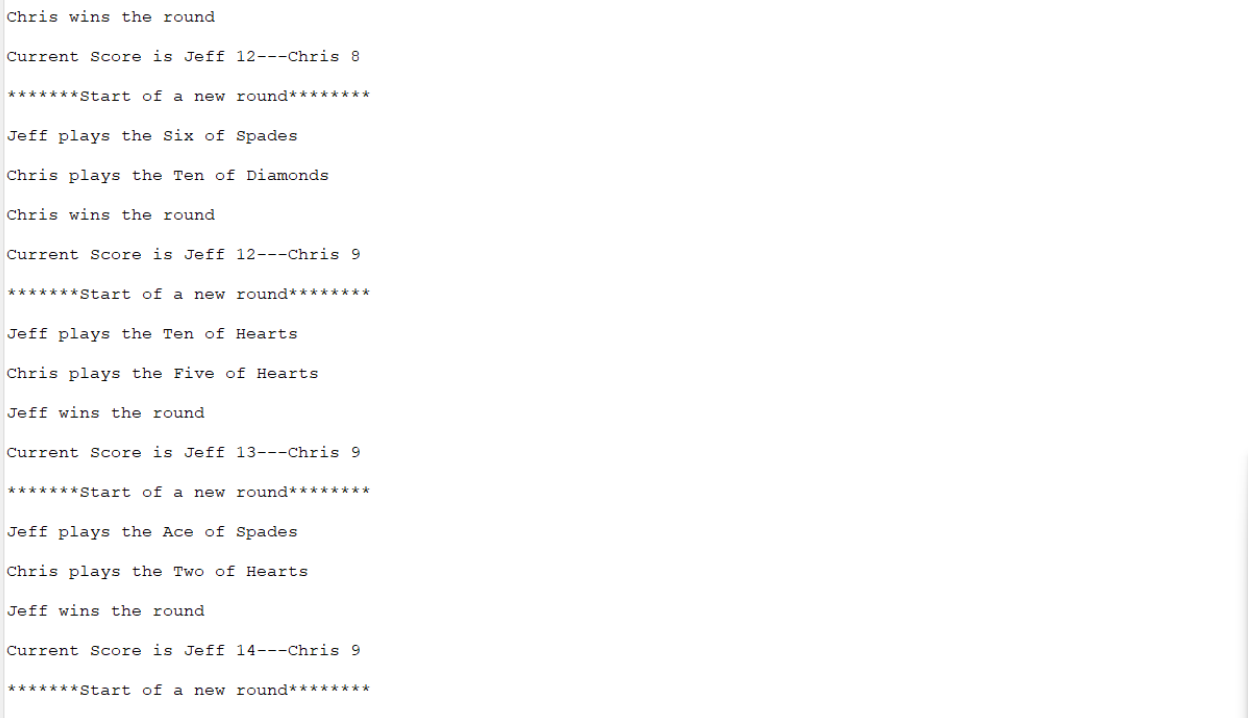
```
Chris plays the Two of Clubs

Jeff wins the round

Current Score is Jeff 9---Chris 5

*******Start of a new round********

Jeff plays the Nine of Clubs

Chris plays the Ace of Hearts

Chris wins the round

Current Score is Jeff 9---Chris 6

*******Start of a new round********

Jeff plays the Nine of Spades

Chris plays the Queen of Diamonds

Chris wins the round

Current Score is Jeff 9---Chris 7

*******Start of a new round********

Jeff plays the Queen of Spades

Chris plays the Nine of Hearts

Jeff wins the round

Current Score is Jeff 10---Chris 7
```



```
*******Start of a new round********

Jeff plays the Four of Diamonds

Chris plays the Two of Spades

Jeff wins the round

Current Score is Jeff 11---Chris 7

*******Start of a new round********

Jeff plays the Three of Diamonds

Chris plays the Three of Clubs

tie

Current Score is Jeff 11---Chris 7

*******Start of a new round********

Jeff plays the Eight of Clubs

Chris plays the Two of Diamonds

Jeff wins the round

Current Score is Jeff 12---Chris 7

*******Start of a new round********

Jeff plays the Queen of Clubs

Chris plays the Ace of Clubs
```



```
*******Start of a new round********

Jeff plays the Ten of Clubs

Chris plays the Six of Diamonds

Jeff wins the round

Current Score is Jeff 15---Chris 9

Jeff is better at war!!
```



```
Chris wins the round

Current Score is Jeff 12---Chris 8

*******Start of a new round********

Jeff plays the Six of Spades

Chris plays the Ten of Diamonds

Chris wins the round

Current Score is Jeff 12---Chris 9

*******Start of a new round********

Jeff plays the Ten of Hearts

Chris plays the Five of Hearts

Jeff wins the round

Current Score is Jeff 13---Chris 9

*******Start of a new round********

Jeff plays the Ace of Spades

Chris plays the Two of Hearts

Jeff wins the round

Current Score is Jeff 14---Chris 9

*******Start of a new round********
```